Prompting LLMs for the Run-Time Event Calculus

Andreas Kouvaras
University of Piraeus, Greece

Periklis Mantenoglou
Örebro University, Sweden

Alexander Artikis
University of Piraeus, Greece
NCSR "Demokritos", Athens, Greece

Abstract

Composite activity recognition systems analyse streams of low-level, symbolic events to identify instances of complex activities based on their formal definitions. Crafting these definitions is a challenging task, as it often requires specifying intricate spatio-temporal constraints, and acquiring labeled data for automated learning is difficult. To address this challenge, we introduce a method that leverages pre-trained Large Language Models (LLMs) to generate composite activity definitions, in the language of the Run-Time Event Calculus, from natural language descriptions.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning

Keywords and phrases Event Calculus, temporal pattern matching, composite event recognition

Digital Object Identifier 10.4230/LIPIcs.TIME.2025.18

Category Short Paper

Supplementary Material *Software*: `https://github.com/aartikis/rtec`

Acknowledgements This work was supported partly by the EU-funded project ENEXA (101070305), and partly by the Wallenberg AI, Autonomous Systems and Software Program funded by the Knut and Alice Wallenberg Foundation.

1 Introduction

Composite event recognition (CER) systems process streams of symbolic, time-stamped events, in order to detect instances of composite activities of interest via temporal pattern matching over these input event streams [17, 7, 8, 1]. In Maritime Situational Awareness (MSA) e.g. the task is to monitor vessel position streams for detecting composite activities such as loitering and violations of the regulations concerning sailing in protected areas [3]. Composite activities are defined in a rigorous manner, by means of a formal pattern specification language. Consider, e.g., CER for city transport management (CTM), where we need to detect composite activities relating to the quality of public transportation based on symbolic event streams that stem from sensor data and contextual information, like an abrupt acceleration or a significant increase in passenger density within a bus. Towards safe public transportation, we would like to detect unsafe driving behaviours. To do this, we may employ a pattern stating that a bus is driven unsafely if it is making sharp turns, or it is accelerating or decelerating abruptly. A sharp turn may be defined as another composite activity, composed of consecutive "change in direction" events, while deriving abrupt acceleration and deceleration events may require background knowledge concerning the type of bus being used, complicating the formal definition of unsafe driving. As a result, constructing a pattern for a composite activity may become quite involved.

Hand-crafting composite activity definitions requires knowledge of formal languages – domain experts, such as transport engineers, cannot be expected to have such knowledge. Moreover, automatically constructing composite activity patterns via specialised machine learning techniques often requires large training datasets with labels for composite activity instances [6, 9, 15]. Unfortunately, the use of such techniques is commonly prohibited due to the scarcity of labels for the inherently infrequent composite activities.

To address these issues, we propose a method that constructs composite activity patterns from natural language descriptions using pre-trained Large Language Models (LLMs). We prompt LLMs to transform composite activity definitions expressed in natural language into logic programming definitions in the language of the "Run-Time Event Calculus" (RTEC) [14, 11, 12]. RTEC is an implementation of the Event Calculus, i.e., a formalism for representing events and reasoning about their effects over time [10, 5, 4]. RTEC is optimised by means of windowing and caching algorithms, demonstrating scalability in challenging domains, such as MSA and CTM, outperforming state-of-the-art CER systems [14, 13]. A key feature of our prompting method is that it is not custom to a single domain, but may be re-used, in a zero-shot manner, for the generation of composite activity definitions in any CER domain.

2 Background: RTEC

RTEC is a formal, logic programming framework that extends the Event Calculus [10] with optimisation techniques for CER [2, 14, 12]. The language of RTEC includes sorts for representing time, events and fluents, i.e., properties whose values may change over time. RTEC employs a linear time-line with non-negative integer time-points. A fluent-value pair (FVP) $F{=}V$ denotes that fluent F has value V. $\mathsf{happensAt}(E, T)$ signifies that event E occurs at time-point T. $\mathsf{initiatedAt}(F = V, T)$ (resp. $\mathsf{terminatedAt}(F = V, T)$) expresses that a time period during which a fluent F has the value V continuously is initiated (terminated) at T. $\mathsf{holdsAt}(F = V, T)$ states that F has value V at T, while $\mathsf{holdsFor}(F = V, I)$ expresses that $F{=}V$ holds continuously in the maximal intervals of list I.

A formalisation of composite activity definitions in RTEC is called *event description*. An event description may contain rules defining two types of FVPs: "simple" and "statically determined". A simple FVP is defined using a set of initiatedAt and terminatedAt rules, and is subject to the commonsense law of inertia, i.e., an FVP $F{=}V$ holds at a time-point T, if $F{=}V$ has been "initiated" by an event at a time-point earlier than T, and not "terminated" by another event in the meantime.

▶ **Example 1** (Within area)**.** In the maritime domain, vessel activity may be disallowed in certain areas, e.g., fisheries restricted areas. Thus, it is desirable to compute the maximal intervals during which a vessel is in such an area. See the definition of a simple FVP below:

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{withinArea}(\mathit{Vessel}, \mathit{AreaType}) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathit{entersArea}(\mathit{Vessel}, \mathit{AreaID}), T), \quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned} \tag{1}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{withinArea}(\mathit{Vessel}, \mathit{AreaType}) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathit{leavesArea}(\mathit{Vessel}, \mathit{AreaID}), T), \quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned} \tag{2}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{withinArea}(\mathit{Vessel}, \mathit{AreaType}) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathit{gapStart}(\mathit{Vessel}), T).\end{aligned} \tag{3}$$

$\mathit{withinArea}(\mathit{Vessel}, \mathit{AreaType})$ is a Boolean fluent denoting that a Vessel is in an area of type $\mathit{AreaType}$, while $\mathit{entersArea}(\mathit{Vessel}, \mathit{AreaID})$, $\mathit{leavesArea}(\mathit{Vessel}, \mathit{AreaID})$ and $\mathit{gapStart}(\mathit{Vessel})$ are input events, derived by the online processing of vessel position signals,

and their spatial relations with areas of interest [16]. $areaType(AreaID, AreaType)$ is an atemporal predicate storing background knowledge regarding the types of areas in a dataset. Rules (1) and (2) state that $withinArea(Vessel, AreaType)$ is initiated (resp. terminated) as soon as a *Vessel* enters (leaves) an area *AreaID*, whose type is *AreaType*. Rule (3) expresses that $withinArea(Vesel, AreaType)$ is terminated when there is a communication gap, i.e., when the *Vessel* stops transmitting its position, and we become uncertain of its whereabouts. ⌟

A statically determined FVP $F = V$ is defined via a rule with head $\mathsf{holdsFor}(F = V, I)$. This rule computes the maximal intervals during which $F = V$ holds continuously by applying a set interval manipulation operations, i.e., union_all, intersect_all and relative_complement_all, on the maximal intervals of other FVPs.

▶ **Example 2** (Anchored and moored vessels)**.** Consider the following definition of a statically determined FVP:

$$\begin{aligned}&\mathsf{holdsFor}(anchoredOrMoored(Vessel) = \mathsf{true}, I) \leftarrow \\ &\quad \mathsf{holdsFor}(stopped(Vessel) = farFromPorts, I_{sf}), \\ &\quad \mathsf{holdsFor}(withinArea(Vessel, anchorage) = \mathsf{true}, I_a), \quad \mathsf{intersect\_all}([I_{sf}, I_a], I_{sfa}), \\ &\quad \mathsf{holdsFor}(stopped(Vl) = nearPorts, I_{sn}), \quad \mathsf{union\_all}([I_{sfa}, I_{sn}], I).\end{aligned} \tag{4}$$

$anchoredOrMoored(Vessel)$ is a Boolean statically determined fluent, defined in terms of three other FVPs: $stopped(Vessel) = farFromPorts$, $stopped(Vessel) = nearPorts$ and $withinArea(Vessel, anchorage) = \mathsf{true}$. The multi-valued fluent $stopped(Vessel)$ expresses the periods during which a *Vessel* is idle near some port or far from all ports. Rule (4) derives the intervals during which a *Vessel* is both stopped far from all ports and within an anchorage area, by applying the intersect_all operation on the lists of maximal intervals I_{sf} and I_a. The output of this operation is list I_{sfa}. Subsequently, list I is derived by applying union_all on lists I_{sfa} and I_{sn}. This way, list I contains the maximal intervals during which a *Vessel* has stopped near some port or within an anchorage area. ⌟

A statically determined FVP holds as long as a Boolean combination of other FVPs is satisfied. Thus, statically determined FVPs are tailored for modeling composite activities that may be defined by applying conjunction, disjunction and negation operators on other activities – see "anchored or moored". "Inertial" composite activities, i.e., activities that persist through time and may arise (or conclude) based on the satisfaction of a set of instantaneous conditions, are expressed using initiatedAt and terminatedAt rules. Typically, a statically determined FVP representation leads to more efficient reasoning, but not all simple FVPs are translatable to statically determined ones. A formal analysis, including an account of the syntax, semantics and reasoning algorithms of RTEC may be found in [14, 12].

3 Prompt Pipeline

Hand-crafting composite activity definitions requires knowledge of the language of RTEC – maritime experts e.g. cannot be expected to have such knowledge. To address this issue, we present a prompting approach that leverages the power of LLMs for constructing composite activity definitions. Figure 1 presents the pipeline for translating natural language descriptions of composite activities into RTEC rules. First, we introduce to the LLM the core predicates of RTEC (**Prompt R**), and provide the syntax for the definitions of simple and statically determined FVPs (**Prompt S**). Then, we proceed with each application domain – for each such domain we present the items of the input stream, i.e. the events and input FVPs (**Prompt E** and **Prompt F**), and the background knowledge predicates (**Prompt B**) – recall e.g. *areaType* in rule-set (1)–(3). Subsequently, **Prompt G** asks the LLM to translate

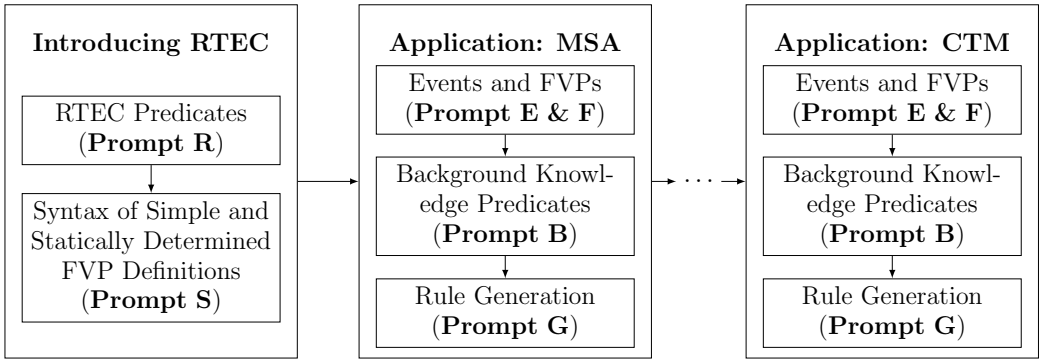

Figure 1 LLM prompting for composite activity definition generation.

a natural language description of each composite activity of the application domain under consideration into a set of RTEC rules. We may customise and repeat **Prompts E, F, B** and **G** for each subsequent application domain. **Prompts R** and **S** are not repeated.

Prompt R introduces the predicates of RTEC. Subsequently, we use **Prompt S** to demonstrate to the LLM how to express a composite activity as a simple or a statically determined FVP. We start with a description of the syntax of the rules expressing simple FVPs. Then, we employ chain-of-thought prompting, according to which we provide natural language descriptions of two example composite activities, and show how these activities may be expressed as simple FVPs. The prompt fragment below illustrates this process; a fragment illustrating statically determined FVPs, is presented in the Appendix. The chosen examples concern MSA. Lines 14, 17 and 20 of the prompt below should be replaced with resp. rules (1), (2), and (3).

Listing 1 Fragment of **Prompt S.**

```
1 There are two ways in which a composite activity may be defined in the language
  of RTEC. In the first case, a composite activity definition may be specified by
  means of rules with ‘initiatedAt(F=V, T)’ or ‘terminatedAt(F=V, T)’ in their head.
   This is called a simple fluent definition.
2
3 The first body literal of an ‘initiatedAt(F=V,T)’ rule is a positive ‘happensAt’
  predicate; this predicate is followed by a possibly empty set of positive or
  negative ‘happensAt’ and ‘holdsAt’ predicates. Negative predicates are prefixed
  with ‘not’ which expresses negation-by-failure. In some cases, the body of an  ‘
  initiatedAt(F=V,T)’ rule may include predicates expressing background knowledge.
4
5 ‘terminatedAt(F=V,T)’ rules are specified in a similar way.
6
7 Below you may find two examples of composite activity definitions, from the
  maritime domain, expressed as simple fluents.
8
9 Example 1: Given a composite maritime activity description, provide the rules in
  the language of RTEC.
10 Composite Maritime Activity Description: ‘withinArea’. This activity starts when
  a vessel enters an area of interest. The activity ends when the vessel leaves the
   area that it had entered, or when the vessel stops transmitting its position,
  since we can no longer assume that the vessel remains in the same area in the
  case of transmission gaps.
11
```

```
Answer:
The activity ‘withinArea’ is expressed as a Boolean simple fluent with two
arguments, i.e., ‘Vessel’ and ‘AreaType’. This activity starts when a vessel
enters an area of interest. We use an ‘initiatedAt’ rule to express this
initiation condition. The body literals of this rules are an event labelled ‘
entersArea’ with two arguments, ’Vessel’ and ‘Area’, and a background knowledge
predicate named ‘areaType’ with two arguments, ‘Area’ and ‘AreaType’. This rule
in the language of RTEC is the following:
<Rule (1)>

The activity ‘withinArea’ ends when a vessel leaves the area that it had entered.
 We use a ‘terminatedAt’ rule to describe this termination condition. This rule
includes an event named ‘leavesArea’ with two arguments, i.e. ‘Vessel’ and ’Area’,
 and the background knowledge predicate ‘areaType’. This rule in the language of
RTEC language is:
<Rule (2)>

In addition to the aforementioned conditions, the activity ‘withinArea’ ends when
 the vessel stops transmitting its position, i.e. when a communication gap starts.
 We use a ‘terminatedAt’ rule to express this termination condition. In this rule,
 the second argument of the ‘withinArea’ fluent is a ‘free’ Prolog variable, i.e.
 a variable starting with ‘_’. The body of this rule includes a single event
named ‘gap_start’ with one argument, i.e. ‘Vessel’. This rule in the language of
RTEC is:
<Rule (3)>

Example 2: <Description 2>
```

According to **Prompt S**, there are two ways in which a natural language description of a composite activity should be expressed. First, the textual description may indicate the conditions in which the composite activity is said to start taking place, as well as the conditions in which the activity is said to stop taking place. This may be achieved with the use of key phrases such as “the activity starts” and “the activity ends” (see **Prompt S** above). Second, we may identify the conditions that must be satisfied so that the composite activity in question holds at any given time. To achieve this, we may enter in the textual description of the activity the key phrase “as long as” (see **Prompt S – Statically determined FVPs** in the Appendix). The first type of textual description implies a simple FVP representation, while the second type of description implies a statically determined FVP formulation. In any case, the compiler of RTEC will choose the most efficient representation [12].

4 Further Work

We aim to evaluate our approach using leading LLMs in diverse domains, such as CER for Maritime Situational Awareness (MSA) and City Transport Management (CTM). Moreover, we aim to fine-tune manageable versions of LLMs for avoiding any errors that may occur in rule generation.

References

1 Elias Alevizos, Alexander Artikis, and Georgios Paliouras. Complex event recognition with symbolic register transducers. *Proc. VLDB Endow.*, 17(11):3165–3177, 2024. doi:10.14778/3681954.3681991.

2 Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015. doi:10.1109/TKDE.2014.2356476.

3 Alexander Artikis and Dimitris Zissis, editors. *Guide to Maritime Informatics*. Springer, 2021. doi:10.1007/978-3-030-61852-0.

4 Iliano Cervesato and Angelo Montanari. A calculus of macro-events: Progress report. In *TIME*, pages 47–58, 2000. doi:10.1109/TIME.2000.856584.

5 Luca Chittaro and Angelo Montanari. Efficient temporal reasoning in the cached event calculus. *Comput. Intell.*, 12:359–382, 1996. doi:10.1111/J.1467-8640.1996.TB00267.X.

6 Lars George, Bruno Cadonna, and Matthias Weidlich. Il-miner: Instance-level discovery of complex event patterns. *Proc. VLDB Endow.*, 10(1):25–36, 2016. doi:10.14778/3015270.3015273.

7 Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020. doi:10.1007/S00778-019-00557-W.

8 Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Trans. Database Syst.*, 46(4):16:1–16:49, 2021. doi:10.1145/3485463.

9 Nikos Katzouris, Georgios Paliouras, and Alexander Artikis. Online learning probabilistic event calculus theories in answer set programming. *Theory Pract. Log. Program.*, 23(2):362–386, 2023. doi:10.1017/S1471068421000107.

10 R. Kowalski and M. Sergot. A logic-based calculus of events. *New Gen. Computing*, 4(1):67–96, 1986. doi:10.1007/BF03037383.

11 Periklis Mantenoglou and Alexander Artikis. Extending the range of temporal specifications of the run-time event calculus. In *TIME*, volume 318, pages 6:1–6:14, 2024. doi:10.4230/LIPICS.TIME.2024.6.

12 Periklis Mantenoglou and Alexander Artikis. Temporal specification optimisation for the event calculus. In *AAAI-25*, pages 15075–15082, 2025. doi:10.1609/AAAI.V39I14.33653.

13 Periklis Mantenoglou, Dimitrios Kelesis, and Alexander Artikis. Complex event recognition with allen relations. In *KR*, pages 502–511, 2023. doi:10.24963/KR.2023/49.

14 Periklis Mantenoglou, Manolis Pitsikalis, and Alexander Artikis. Stream reasoning with cycles. In *KR*, pages 544–553, 2022.

15 Evangelos Michelioudakis, Alexander Artikis, and Georgios Paliouras. Online semi-supervised learning of composite event rules by combining structure and mass-based predicate similarity. *Mach. Learn.*, 113(3):1445–1481, 2024. doi:10.1007/S10994-023-06447-1.

16 Georgios M. Santipantakis, Akrivi Vlachou, Christos Doulkeridis, Alexander Artikis, Ioannis Kontopoulos, and George A. Vouros. A stream reasoning system for maritime monitoring. In *TIME*, volume 120, pages 20:1–20:17, 2018. doi:10.4230/LIPICS.TIME.2018.20.

17 Walker M. White, Mirek Riedewald, Johannes Gehrke, and Alan J. Demers. What is "next" in event processing? In *PODS*, pages 263–272, 2007. doi:10.1145/1265530.1265567.

A Prompt S

Below we present a fragment of **Prompt S** that introduces the syntax and an example of statically determined FVPs.

Listing 2 Prompt S – Statically determined FVPs.

```
The second way in which a composite activity may be defined in the language of
RTEC concerns statically determined fluents. In this case, a composite activity
definition may be specified by means of a rule with 'holdsFor(F=V, I)' in its
head. The body of such a rule may include 'holdsFor' conditions for fluents other
 than F, as well as some of the interval manipulation constructs of RTEC, i.e. '
union_all', 'intersect_all', and 'relative_complement_all'. In some cases, a '
holdsFor(F=V, I)' rule may include predicates expressing background knowledge. A
rule with 'holdsFor(F=V, I)' in the head is called a statically determined fluent
 definition. Below you may find two examples of composite maritime activities
expressed as statically determined fluents.
```

```
2
3 Example 1: Given a composite maritime activity description, provide the rules in
  the language of RTEC.
4 Composite Maritime Activity Description: ‘underWay’. This activity lasts as long
  as a vessel is not stopped.
5
6 Answer: The activity ‘underWay’ is expressed as a statically determined fluent.
  Rules with ‘holdsFor’ in the head specify the conditions in which a fluent holds.
   We use a ‘holdsFor’ rule to describe that the ‘underWay’ activity lasts as long
  as a vessel is not stopped. The output is Boolean fluent named ‘underWay’ with
  one argument, i.e. ‘Vessel’. We specify ‘underWay’ with the use of the fluent ‘
  movingSpeed’. We express ‘underWay’ as the disjunction of the three values of ‘
  movingSpeed’, i.e. ‘below’, ‘normal’ and ‘above’. Disjunction in ‘holdsFor’ rules
   is expressed by means of ‘union_all’. This rule is expressed in the language of
  RTEC as follows:
7
8 holdsFor(underWay(Vessel)=true, I) :-
9         holdsFor(movingSpeed(Vessel)=below, I1),
10        holdsFor(movingSpeed(Vessel)=normal, I2),
11        holdsFor(movingSpeed(Vessel)=above, I3),
12        union_all([I1,I2,I3], I).
13
14 Example 2: <Description 2>
```